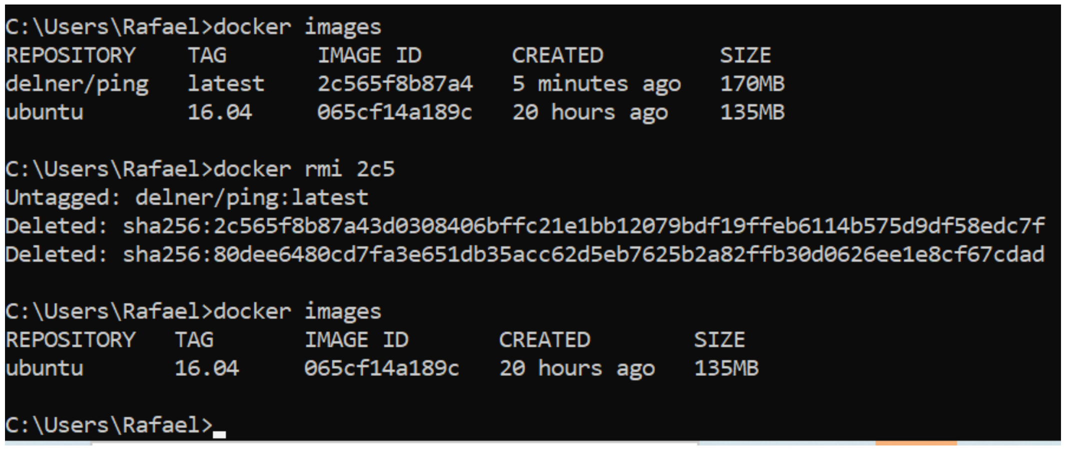
Maceda, Rafael
2016-01337
CoE 197S

Feature Activity 03

A. Getting set-up

Deleted image made from last activity

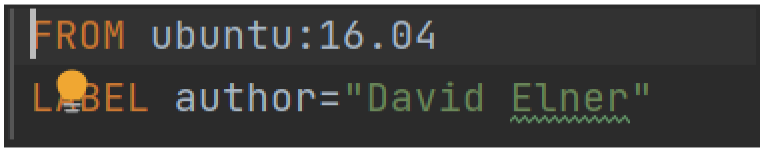


```
C:\Users\Rafael>docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
delner/ping   latest    2c565f8b87a4   5 minutes ago    170MB
ubuntu        16.04     065cf14a189c   20 hours ago     135MB

C:\Users\Rafael>docker rmi 2c5
Untagged: delner/ping:latest
Deleted: sha256:2c565f8b87a43d0308406bffc21e1bb12079bdf19ffeb6114b575d9df58edc7f
Deleted: sha256:80dee6480cd7fa3e651db35acc62d5eb7625b2a82ffb30d0626ee1e8cf67cdad

C:\Users\Rafael>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       16.04     065cf14a189c   20 hours ago   135MB

C:\Users\Rafael>_
```

B. Creating a Dockerfile

```
1. Create a new file named `Dockerfile` in your working
directory:

  ```
    $ touch Dockerfile
    $
  ```

2. Open `Dockerfile` with your favorite text editor.

2. Inside the file, we'll need to add some important headers.

   The `FROM` directive specifies what base image this new
image will be built upon. (Ubuntu in our case.)

   The `LABEL` directive adds a label the image. Useful for
adding metadata.

   Add the following two lines to the top of your file:

  ```
   FROM ubuntu:16.04
   LABEL author="David Elner"
```
```

~~~
```
~~~



```
FROM ubuntu:16.04
LABEL author="David Elner"
```

3. Then we'll need to add some commands to modify the image.

   The `RUN` directive runs a command inside the image, and rolls any changes to the filesystem into a commit. A typical Dockerfile will contain several `RUN` statements, each committing their changes on top of the previous.

   To install `ping`, we'll need to run `apt-get update` and `apt-get install`. First, add the `apt-get update` command:

   ```
   RUN apt-get update
   ```

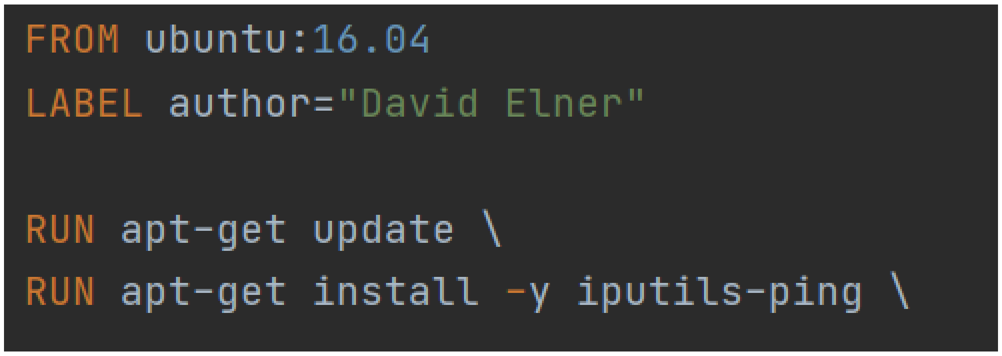
   Then add the `apt-get install` command:

   ```
   RUN apt-get install -y iputils-ping
   ```

   Notice we added the `-y` flag. When building Docker images, these commands will run non-interactively. Normally the `apt-get` command will prompt you "Y/n?" if you want to proceed. The `-y` flag avoids that prompt by always answering "Y".

   Our file should now look something like this:

   ```
   FROM ubuntu:16.04
   LABEL author="David Elner"

   RUN apt-get update

   RUN apt-get install -y iputils-ping
   ```



```
FROM ubuntu:16.04
LABEL author="David Elner"

RUN apt-get update \
RUN apt-get install -y iputils-ping \
```

And with that, we should be ready to build our image.

C. Building the Dockerfile

To build Docker images from Dockerfiles, we use the `docker build` command. The `docker build` command reads a Dockerfile, and runs its instructions to create a new image.

1. Let's build our image.

Running the following builds and tags the image:

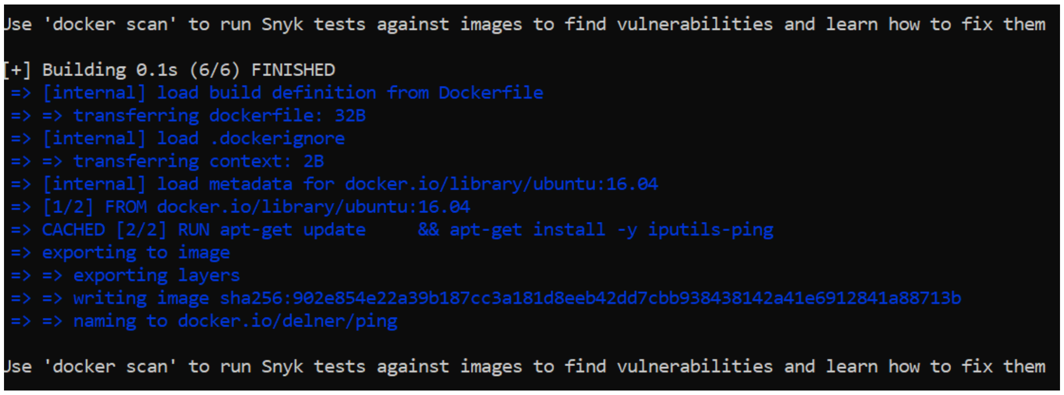
```
$ docker build -t 'delner/ping' .
```

The use of `.` in the arguments is significant here. `docker build` looks for files named `Dockerfile` by default to run. So by giving it a `.`, we're telling Docker to use the `Dockerfile` in our current directory. If this Dockerfile was actually named anything else, you'd change this to match the name of the file.

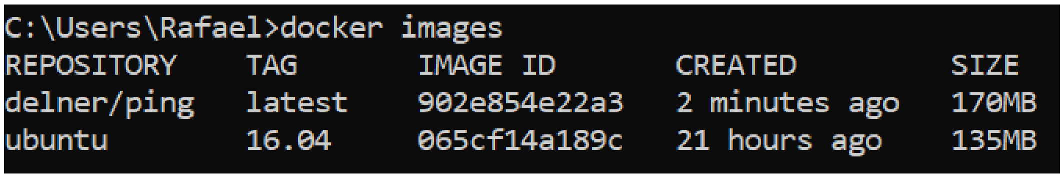
Also notice the output about "steps" here. Each directive in your Dockerfile maps to a step here, and after each step is completed, it becomes a commit. Why does this matter though?

Docker layers each commit on top of the other, like an onion. By doing so, it can keep image sizes small, and when rebuilding images, it can even reuse commits that are unaffected by changes to make builds run quicker.

We can see this caching behavior in action if we simply rerun the same command again:


```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

[+] Building 0.1s (6/6) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/ubuntu:16.04
 => [1/2] FROM docker.io/library/ubuntu:16.04
 => CACHED [2/2] RUN apt-get update     && apt-get install -y iputils-ping
 => exporting to image
 => => exporting layers
 => => writing image sha256:902e854e22a39b187cc3a181d8eeb42dd7cbb938438142a41e6912841a88713b
 => => naming to docker.io/delner/ping

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

Running `docker images`, we can see our newly built image.


```
C:\Users\Rafael>docker images
REPOSITORY     TAG        IMAGE ID       CREATED          SIZE
delner/ping    latest     902e854e22a3   2 minutes ago    170MB
ubuntu         16.04      065cf14a189c   21 hours ago     135MB
```

D. Optimizing the Dockerfile

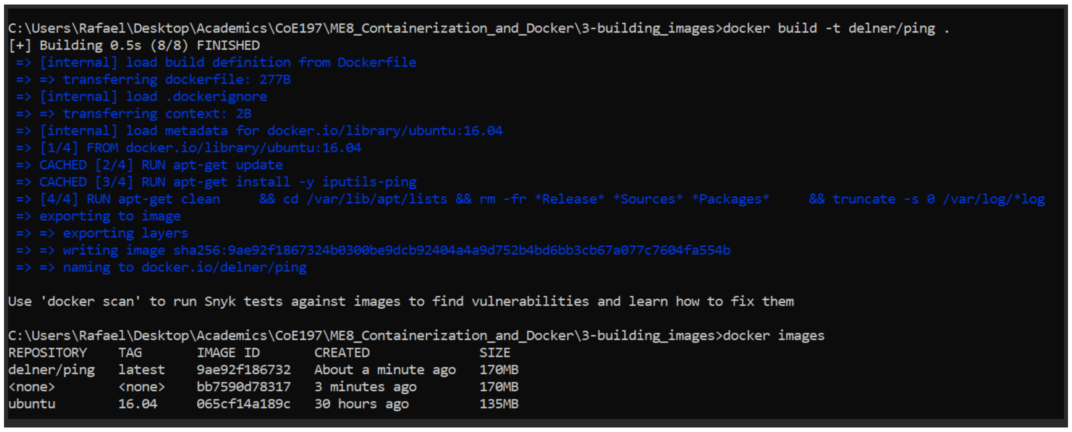
Looking at new image, we can see it is `159MB` in size versus its base image of `117MB`. That's a pretty big change in size for installing some utilities. That will take more disk space and add additional time to pushes/pulls of this image.

But why is it that much bigger? The secret is in the `RUN` commands. As mentioned before, any filesystem changes are committed after the `RUN` command completes. This includes any logs, or temporary data written to the filesystem which might be completely inconsequential to our image.

In our case, the use of `apt-get` generates a lot of this fluff we don't need in our image. We'll need to modify these `RUN` directives slightly.

We can start with removing any old logs after the install completes. Adding the following to the b

Then running `build` again yields…



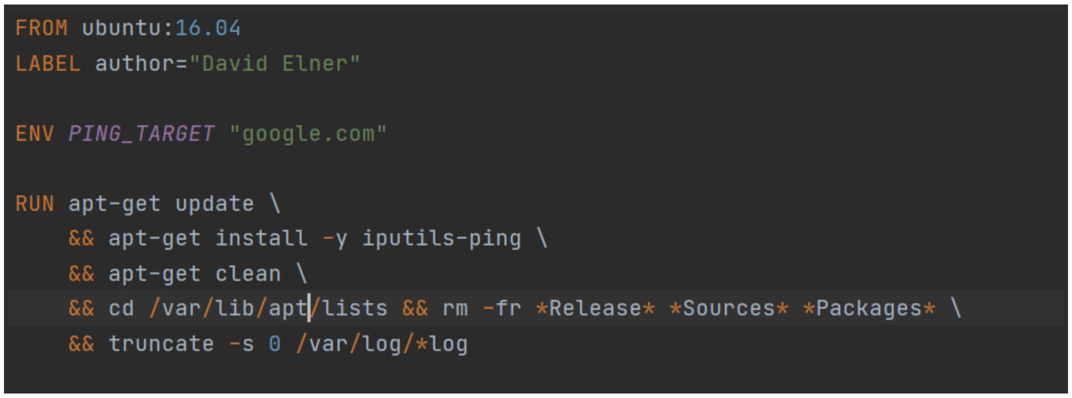
```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker build -t delner/ping .
[+] Building 0.5s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 277B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/ubuntu:16.04
 => [1/4] FROM docker.io/library/ubuntu:16.04
 => CACHED [2/4] RUN apt-get update
 => CACHED [3/4] RUN apt-get install -y iputils-ping
 => [4/4] RUN apt-get clean     && cd /var/lib/apt/lists && rm -fr *Release* *Sources* *Packages*     && truncate -s 0 /var/log/*log
 => exporting to image
 => => exporting layers
 => => writing image sha256:9ae92f1867324b0300be9dcb92404a4a9d752b4bd6bb3cb67a077c7604fa554b
 => => naming to docker.io/delner/ping

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
delner/ping   latest    9ae92f186732   About a minute ago   170MB
<none>        <none>    bb7590d78317   3 minutes ago        170MB
ubuntu        16.04     065cf14a189c   30 hours ago         135MB
```

...yields no improvement? What's going on here?

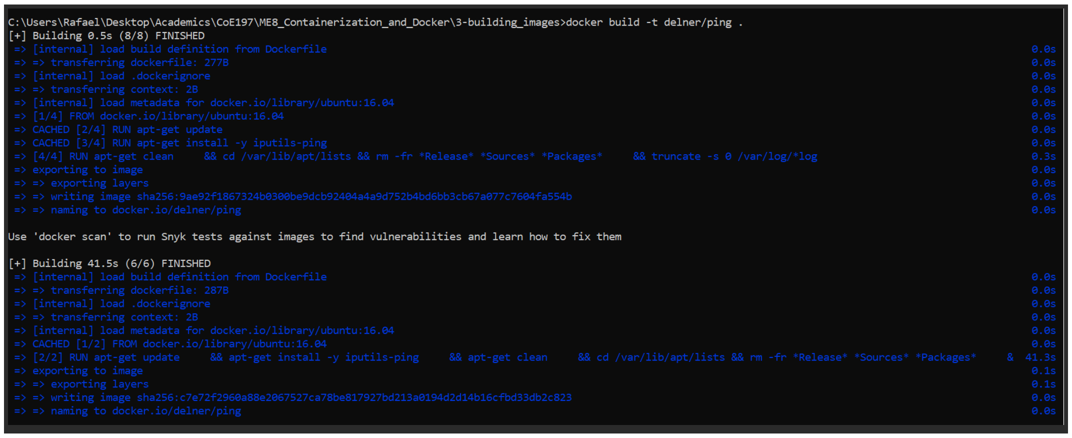
Turns out because how commits are layered one upon the other, if there's fluff hanging around from a previous commit, it won't matter if you clean it up in a future `RUN` directive. It will be permanently apart of the history, thus the image size.

Our fluff actually happens to originate from the `apt-get update` command, which leaves a bunch of package lists around that we don't need. The easiest way to deal with this is to collapse all of the related `RUN` directives together.

The rewritten Dockerfile should like:

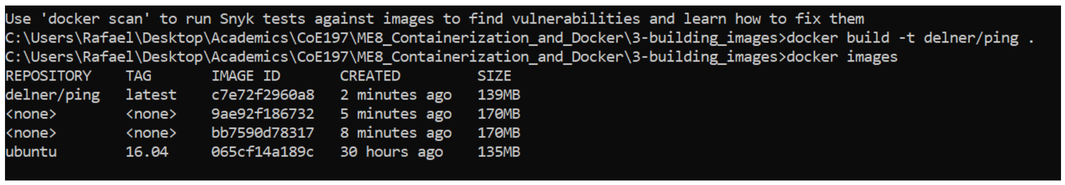


```
FROM ubuntu:16.04
LABEL author="David Elner"

ENV PING_TARGET "google.com"

RUN apt-get update \
    && apt-get install -y iputils-ping \
    && apt-get clean \
    && cd /var/lib/apt/lists && rm -fr *Release* *Sources* *Packages* \
    && truncate -s 0 /var/log/*log
```



```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker build -t delner/ping .
[+] Building 0.5s (8/8) FINISHED
 => [internal] load build definition from Dockerfile                                                                                                    0.0s
 => => transferring dockerfile: 277B                                                                                                                    0.0s
 => [internal] load .dockerignore                                                                                                                       0.0s
 => => transferring context: 2B                                                                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:16.04                                                                                         0.0s
 => [1/4] FROM docker.io/library/ubuntu:16.04                                                                                                           0.0s
 => CACHED [2/4] RUN apt-get update                                                                                                                     0.0s
 => CACHED [3/4] RUN apt-get install -y iputils-ping                                                                                                    0.0s
 => [4/4] RUN apt-get clean     && cd /var/lib/apt/lists && rm -fr *Release* *Sources* *Packages*     && truncate -s 0 /var/log/*log                    0.3s
 => exporting to image                                                                                                                                  0.0s
 => => exporting layers                                                                                                                                 0.0s
 => => writing image sha256:9ae92f1867324b0300be9dcb92404a4a9d752b4bd6bb3cb67a077c7604fa554b                                                            0.0s
 => => naming to docker.io/delner/ping                                                                                                                  0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

[+] Building 41.5s (6/6) FINISHED
 => [internal] load build definition from Dockerfile                                                                                                    0.0s
 => => transferring dockerfile: 287B                                                                                                                    0.0s
 => [internal] load .dockerignore                                                                                                                       0.0s
 => => transferring context: 2B                                                                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:16.04                                                                                         0.0s
 => CACHED [1/2] FROM docker.io/library/ubuntu:16.04                                                                                                    0.0s
 => [2/2] RUN apt-get update     && apt-get install -y iputils-ping     && apt-get clean     && cd /var/lib/apt/lists && rm -fr *Release* *Sources* *Packages*     &  41.3s
 => exporting to image                                                                                                                                  0.1s
 => => exporting layers                                                                                                                                 0.1s
 => => writing image sha256:c7e72f2960a88e2067527ca78be817927bd213a0194d2d14b16cfbd33db2c823                                                            0.0s
 => => naming to docker.io/delner/ping                                                                                                                  0.0s
```

Since our Dockerfile is build for `ping`, let's add the `ENV` and `CMD` directives.

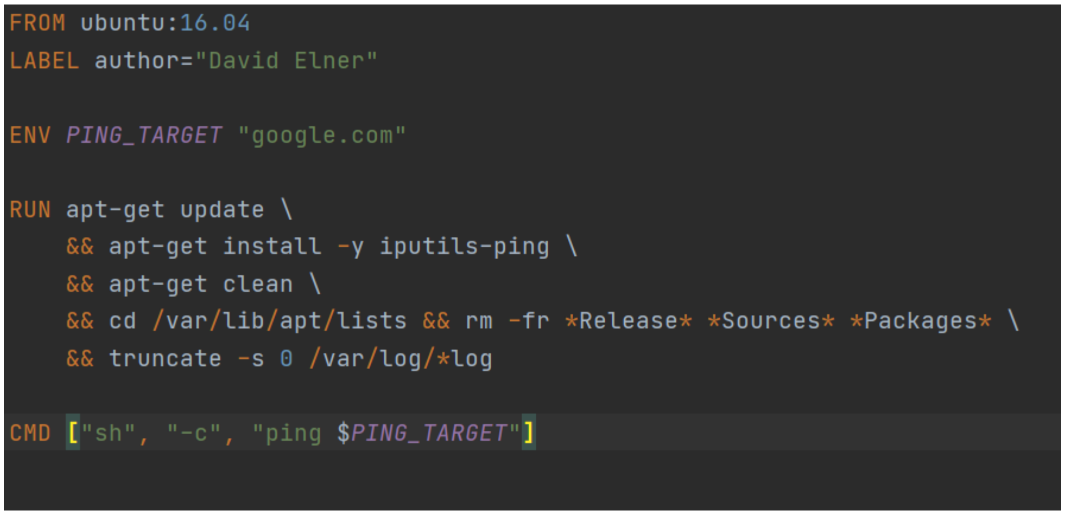


```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker build -t delner/ping .
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
delner/ping   latest    c7e72f2960a8   2 minutes ago   139MB
<none>        <none>    9ae92f186732   5 minutes ago   170MB
<none>        <none>    bb7590d78317   8 minutes ago   170MB
ubuntu        16.04     065cf14a189c   30 hours ago    135MB
```

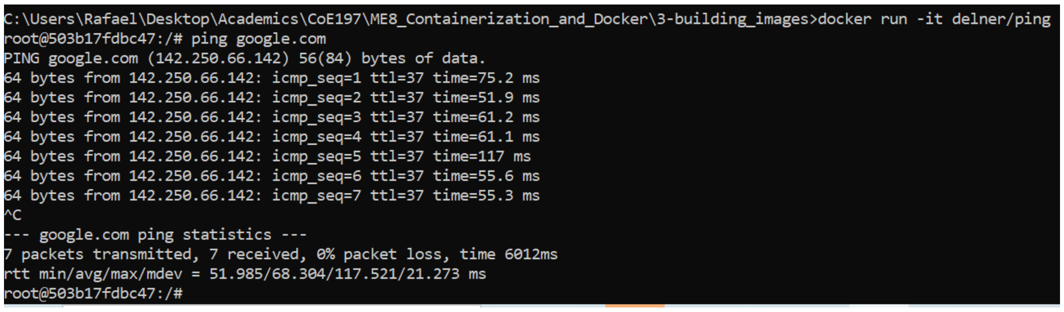
New image is 139 mb, only 4 mb larger

E. Other Dockerfile directives

```
### Other Dockerfile directives

There are [many other useful directives](https://docs.docker.com/engine/reference/builder/) available in the Dockerfile.

Some important ones:

- `COPY`: Copy files from your host into the Docker image.
- `WORKDIR`: Specify a default directory to execute commands from.
- `CMD`: Specify a default command to run.
- `ENV`: Specify a default environment variable.
- `EXPOSE`: Expose a port by default.
- `ARG`: Specify a build-time argument (for more configurable, advanced builds.)

Since our Dockerfile is build for `ping`, let's add the `ENV` and `CMD` directives.
```



```
FROM ubuntu:16.04
LABEL author="David Elner"

ENV PING_TARGET "google.com"

RUN apt-get update \
    && apt-get install -y iputils-ping \
    && apt-get clean \
    && cd /var/lib/apt/lists && rm -fr *Release* *Sources* *Packages* \
    && truncate -s 0 /var/log/*log

CMD ["sh", "-c", "ping $PING_TARGET"]
```



```
C:\Users\Rafael\Desktop\Academics\CoE197\ME8_Containerization_and_Docker\3-building_images>docker run -it delner/ping
root@503b17fdbc47:/# ping google.com
PING google.com (142.250.66.142) 56(84) bytes of data.
64 bytes from 142.250.66.142: icmp_seq=1 ttl=37 time=75.2 ms
64 bytes from 142.250.66.142: icmp_seq=2 ttl=37 time=51.9 ms
64 bytes from 142.250.66.142: icmp_seq=3 ttl=37 time=61.2 ms
64 bytes from 142.250.66.142: icmp_seq=4 ttl=37 time=61.1 ms
64 bytes from 142.250.66.142: icmp_seq=5 ttl=37 time=117 ms
64 bytes from 142.250.66.142: icmp_seq=6 ttl=37 time=55.6 ms
64 bytes from 142.250.66.142: icmp_seq=7 ttl=37 time=55.3 ms
^C
--- google.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6012ms
rtt min/avg/max/mdev = 51.985/68.304/117.521/21.273 ms
root@503b17fdbc47:/#
```